IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of

NEDBAL
Atty. Ref.:
550-320

Serial No.
10/092,424
Group:

Filed:
March 7, 2002
Examiner:

For:
MANAGING COMPUTER PROGRAM CONFIGURATION

* * * * * * * * * * * *

March 15, 2005

Assistant Commissioner for Patents
Washington, DC 20231

Sir:

DECLARATION UNDER RULE 131
SWEARING BEHIND PRESLEY REFERENCE
US 2003/0105838 A1 FILED NOVEMBER 30, 2001

I, Manuel Nedbal, declare as follows:

1. That I am the inventor of the above-identified application filed March 7, 2002;

2. That I conceived of the invention disclosed and claimed in the above application in the United Kingdom, a WTO country, prior to November 30, 2001, the filing date of US Patent Application Publication US 2003/0105838 A1 published Jun 5, 2003 in the name of Darryl Lee Presley (hereinafter "Presley");

3. Exhibit 1 attached hereto is a copy of a patent application (any identifying dates have been redacted) that was prepared on my behalf and that I reviewed prior to November 30, 2001, said application containing a detailed description of the present invention (hereinafter the "prior application").

4. A comparison of the present application and the prior application will show that all of the prior application is included in the present application and the claims in the prior application mirror the claims in the present application.

5. I received and approved the prior application and I understand that it was filed in the US Patent Office prior to November 30, 2001.

6. That I do not know and do not believe that the invention disclosed and claimed in the above-identified application has been in public use or on sale in the United States or patented or described in a printed publication in the United States or any country foreign thereto for more than one year prior to the above application filing date, and that I have never abandoned the invention described and claims in the above application;

7. That all statements made herein of our own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of

the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issuing thereon.

Manuel Nedbal — Date: 15-MAR-2005

Manuel Nedbal

Enclosed:
Exhibit 1

P12469US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION PAPERS

OF

MANUEL NEDBAL

FOR

MANAGING COMPUTER PROGRAM CONFIGURATION DATA

EXHIBIT 1

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of data processing systems. More particularly, this invention relates to the management of computer program configuration data, such as, for example, Windows Registry data, within data processing systems.

Description of the Prior Art

It is known to associate configuration data with computer programs. As an example, when a computer program is first installed a user may be allowed to specify certain options as to how that computer program is to be configured. One example would be the language to be used for the display of messages or instructions, parameters specifying the location of different components of the computer program product and the like. Within the current range of Windows operating system programs, this configuration data is normally stored within the Windows Registry. The structure and content of the Windows Registry can be complex and problems within this data can lead to unreliability of the associated computer programs. Furthermore, it is difficult to reliably and efficiently centrally manage such configuration data installed on computer networks and the like. If System Administrators wish to check if the configuration of the computers on their network is correct and depending on the result of this check want to apply a certain configuration, then they may have little choice but to make this check and change for each separate computer and each configuration setting.

SUMMARY OF THE INVENTION

Viewed from one aspect the present invention provides a computer program product for controlling a computer to validate program configuration data, said computer program product comprising:

comparing code operable to compare an XML data representation of said program configuration data with data defining valid program configuration data;

wherein, if said XML data representation does match said data defining valid program configuration data, then triggering code is operable to trigger a valid program configuration response.

The invention recognises and exploits that program configuration data generally has a regular and ordered structure and well defined ranges of valid configuration parameters such that the program configuration data can be well represented as XML data. Once the configuration data is represented as XML data, then existing tools and resources that are provided for the validation of XML data may be applied to the representation of the configuration data as XML data in order to validate that configuration data. This provides a low overhead, flexible and reliable technique for managing program configuration data.

It will be appreciated that the program configuration data could take a wide variety of forms. As previously discussed, the program configuration data could be registry data associated with the operating system of the computer concerned. Alternatively, program initialisation data, such as "ini" data could form the configuration data. Another possibility is that the configuration data may in its native form be XML data. Generally any configuration store whose data can be mapped into XML data can be used.

It will be appreciated that when the configuration data is not XML data in its native form, then it is necessary to provide a mapping mechanism for mapping the program configuration data into an XML data representation of that program configuration data and vice versa. Thus, live existing configuration data may be mapped and compared to establish validity and for other management reasons.

The data defining valid program configuration data could take a variety of forms, but particularly preferred embodiments use XSD data or DTD data for this purpose. The tools and skills for validating XML data using these schemas are already provided and can be exploited in this way.

Whilst it would be possible to provide special purpose mechanisms for comparing the XML data representation with the validity data, preferred embodiments use an XML parser, which will typically already be provided within a computer system for validating XML data from other sources and for other reasons. This advantageously reduces the overhead required to exploit the above techniques.

In preferred embodiments of the invention the mapping between the program configuration data and the XML data representation takes place via a DOM data representation, which is a common way of holding and manipulating XML data in the computer's main memory. Making this mapping via this intermediate form allows the tools, mechanisms and the skills normally provided for working with DOM data to be reused for the purpose of managing, and in particular editing and adjusting program configuration data, whilst keeping within an overall arrangement that allows rigorous validation of the results of such editing or modification.

It will be appreciated that the validation of the configuration data could take place at a variety of different points within an overall system. In some embodiments it may be desirable to validate the configuration data within a program configuration managing computer before such configuration data is sent out to an associated managed computer. Alternatively and/or additionally the managed computer may operate to validate configuration data that it receives in accordance with the above described techniques.

Viewed from another aspect the present invention provides a computer program product for providing program configuration data for a computer, said computer program product comprising:

receiving code operable to receive an XML data representation of said program configuration data at said computer;

mapping code operable to map between said XML data representation and said program configuration data; and

configuration code operable to at least one of apply said program configuration data to said computer and retrieve said program configuration data from said computer.

This aspect of the invention principally relates to a managed computer which receives configuration data generically specified as XML data and then maps this to the native form for that managed computer before applying that configuration data to the managed computer. Mapping in the opposite direction is also enabled. The XML data can be validated upon receipt and/or before it is sent. This provides a robust and

flexible mechanism for distributing and validating program configuration data and allowing changes to that data as appropriate.

Other aspects of the invention provide methods and apparatus operating in accordance with the above described techniques.

The above, and other objects, features and advantages of this invention will be apparent from the following detailed description of illustrative embodiments which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 schematically represents Windows Registry data that specifies a computer program configuration;

Figure 2 illustrates a representation of DOM data as made accessible in memory corresponding to the configuration data of Figure 1;

Figure 3 is an XML data representation of the configuration data of Figure 1;

Figure 4 is a first representation of an XSD schema corresponding to the XML data of Figure 3;

Figure 5 is a second representation of an XSD schema corresponding to the XML data of Figure 3;

Figure 6 is a flow diagram schematically illustrating the mapping of configuration data to XML data and then the validation of this XML data;

Figure 7 schematically illustrates an application program checking its own configuration data;

Figure 8 schematically illustrates the editing of configuration data and its validation by an application program;

Figure 9 schematically illustrates the technique of Figure 8 followed by the transfer of the validated XML data to a managed computer and the application of that XML data to the managed computer;

Figure 10 illustrates a modification of the technique of Figure 9 in which the managed computer also checks the received XML data again to validate it before it is applied; and

Figure 11 is a schematic diagram representing a general purpose computer of the type that may be used to implement the above described techniques.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

Figure 1 schematically represents a portion of the program configuration data associated with an application program that is stored within the Windows Registry of a computer using the Windows operating system provided by Microsoft Corporation. This data is illustrated as being accessed and viewed via the Registry editer tool that is conventionally provided. It will be seen that the configuration data specifies parameters such as the version of the computer program installed, devices associated with that installation, the language to be used in particular circumstances with associated messages for display in response to particular events that may occur during execution and the like. The specification of computer program configuration data within the Windows Registry in this way is well known within the field of computer programming and will not be described herein any further.

Figure 2 schematically illustrates the configuration data of Figure 1 which has been mapped into DOM data and is being viewed by an associated viewing tool. The mapping of the configuration data into DOM data can be achieved in a wide variety of different ways. One example, is to map keys within the Windows Registry data to complex data types within the DOM data. Values within the Windows Registry can be mapped to simple types within the DOM data. In order to deal efficiently with keys and values within the Windows Registry that can occur any number of times the mapping mechanism may parse the Windows Registry to identify such keys and types and then associate attributes with the different instances of the keys and types encountered. This helps to provide a more efficient and compact DOM representation

of the configuration data which will utlimately result in a more compact and efficient XML representation, with associated XSD validating data.

A table illustrating one example of how Windows Registry configuration data may be mapped in accordance with the present technique is given below.

Registry Data	XSD Data	XML Data	Comments
Values correspond to XML elements			
"valuename"="stringvalue"	<xs:element name="valuename" type="xs:string"/>	<valuename>stringvalue</valuename>	*REG_MULTI_SZ strings can be mapped into lists of XML strings and the other way round.*
"valuename"=dword:dwordvalue	<xs:element name="valuename" type="xs:unsignedLong" />	<valuename>dwordvalue</valuename>	*Conversions between hexadecimal and decimal forms have to be taken into consideration here.*
"valuename"=hex:hexvalue	<xs:element name="valuename" type="xs:hexBinary"/>	<valuename>hexvalue</valuename>	
Keys correspond to XML complex types			
[keyname]	<xs:complexType name="keyname"><xs:all> *... All subkeys and values ...* </xs:all></xs:complexType>	<keyname> *... All subkeys and values ...*</keyname>	*<xs:all> means that this type's sequence of its members does not matter (in contrast to <xs:sequence>)*
Enumeration of Values, which may occur any number of times			
"valuename id=valueid"=" somevalue "	<xs:element name="valuename" Id="valueId" type="sometype"> somevalue </xs:element>	<valuename id="valueid" type="sometype"> somevalue </valuename>	*Changing the format of the registry value name simplifies the XSD validation. The name contains the additional information "valueid", which distinguishes each value from the others.*
Enumeration of Keys, which may occur any number of times			
[keyname id="keyid"]	<xs:complexType name="keyname" id="keyid"><xs:all> *... All subkeys and values ...* </xs:all></xs:complexType>	<keyname id="keyid"> *... All subkeys and values ...*</keyname>	*Similar to the Enumeration of Values, except that any kind and number of subkeys and -values are allowed.*

Known commercially available tools may be used to map from the DOM data as represented in Figure 2 to corresponding XML data as illustrated in Figure 3. The

highly flexible and robust nature of XML as a representation of data will be appreciated from Figure 3. Furthermore, a comparison of the Windows Registry data of Figure 1 stored in its heirarchical structure will be appreciated to map well to the heirarchical data representation that is associated with XML. The recognition and exploitation of the fact that configuration data has a form and structure that is well suited to representation by XML data is an important feature of the current technique and enables many existing tools and resources provided for general XML data manipulation and validation to be reused for the manipulation, management and validation of program configuration data once this is transformed into the form of XML data.

As well as the tools for mapping configuration data into DOM data and XML data, the validation technique described herein also requires associated validation data in the form of XSD data against which XML data may be checked. This XSD data will normally be generated by the provider of the computer program product which its having its configuration managed in accordance with this technique or it can be generated by programs knowing the configuration data of another program. An efficient way to generate this XSD data is to start with a known valid set of Windows Registry configuration data for the computer program concerned. Once this has been mapped into XML data using the above described mapping technique, a tool such as XMLSpy may be aplied to that example XML data representation of the configuration to produce an associated XSD validation template. Figure 4 illustrates one view of the XSD data that may be generated from the XML data previously discussed using such an automated tool. Such an automated tool typically will not provide the most elegant and compact XSD data corresponding to the XML representation of the configuration data. Accordingly, once the tool has produced the XSD data shown in accordance with Figure 4, it is proposed that this XSD data would then be examined and hand edited by a programmer familiar with the application concerned. Such hand editing will typically provide a more compact and realistic XSD representation of the configuration data as well as allowing ranges of valid configuration parameters to be specified in a manner generalised from the particular configuration parameters that may be picked up from the example configuration that was converted using the automated tool. This hand editing of XSD data is a general technique used by those familiar with the technical field and will not be described further.

Figure 5 illustrates an example of XSD data that may be associated with the previously described configuration data and has been generated by hand editing of an automatically provided XSD representation of that XML data.

It will be appreciated that the technique of the present invention is not restricted to the mechanisms for generating associated XSD data as described above nor the particular form of configuration validating data represented by XSD data. However, these techniques are strongly preferred as they do enable the reuse of overhead, resources and skills that are generally already provided.

Figure 6 is a flow diagram illustrating the validation of program configuration data. Windows Registry data 2 is mapped into DOM data 4 by a mapping function 6. This mapping function may operate using the example mappings described above, or alternative mappings that may be possible. The DOM data 4 is streamed into XML data 8 by step 10. This streaming may also be referred to as serialisation. The XML data 8 together with previously generated and associated XSD data 12 is then provided to an XML parser 14 where the XML data 8 is validated against the XSD data to produce a validation result 16. The XML parsers and validation mechanisms are typically already provided within internet browsers and the like which are installed on computers for reasons other than the validation of configuration data.

Figure 7 is a flow diagram illustrating an application program validating its configuration. An application program 18 is provided with an associated set of XSD data 20 by the provider of that application program 18. This XSD data 20 can be viewed as a template for valid configuration data. The provider of the application program 18 will use their knowledge and expertise relating to that application program 18 in order to provide a generic and robust set of XSD data.

The Windows Registry data 22 corresponding to the particular installation concerned is read and mapped into DOM data 24 before being serialised into corresponding XML data 26 which represents the configuration data (Windows Registry data 22). A call may be then made to an XML parser 28 which validates the XML data of the particular installation against the XSD data 20 provided in

association with the application program in order to generate a validation result 30. It will be appreciated that the validation result 30 may be used to trigger either an invalid configuration response, such as generation of an appropriate error message, or a valid configuration response, such, for example, as starting execution of the associated application program.

Figure 8 is a flow diagram schematically illustrating the editing of configuration data using the present techniques. It will be seen that a number of the steps in Figure 8 correspond to those in Figure 7 and these will not be described again. Compared with Figure 7, the DOM data 24 is made available to other applications which may modify that DOM data 24 to customise it or edit it in other ways. An editing application 32 may be used to hand edit the DOM data 24. Alternatively and/or additionally, XML data 34 may be de-serialised and appended to or inserted within the DOM data 24 in a manner to extend the configuration data. Once the editing of the DOM data 24 has been completed, this edited DOM data 24 is serialised into XML data 36 which is then validated against the corresponding XSD data 20. If the validation step is passed at step 38, then the modified configuration data represented by the XML data 36 is mapped back into DOM data and then Windows Registry data 40 for use within the computer concerned to replace the original Windows Registry data 42.

Figure 9 illustrates the process of Figure 8 but in this case the successfully validated XML data is transmitted to another computer (a managed computer) at step 44. The received XML data 46 is de-serialised within the managed computer to form DOM data 48 which is then in turn mapped into Windows Registry data 50 for controlling the configuration of a different instance of the application program concerned that is installed within the managed computer. It will be appreciated that the editing and validation of the configuration data which occurred in the first portion of Figure 9 is carried out by a configuration managing computer, such as a computer operated by a System Administrator, and once this edited program configuration has been successfully validated it is automatically distributed to associated managed computers and applied to those managed computers. In the example of Figure 9, the managed computer does not itself recheck the validity of the configuration data which it receives. Instead it receives this data in the form of an XML data representation of

its configuration which it maps into the required native configuration data and then applies this native configuration data.

Figure 10 illustrates a modification of the technique of Figure 9. After the XML data 44 representing the configuration has been transmitted (transferred) to the managed computer, the managed computer uses its own copy of the application program 52 concerned to read XSD data associated with the configuration data such that this may be validated by the managed computer itself before being applied. In this particular instance, the XML data representation of the configuration is validated both within the configuration managing computer and the managed computer. It would be possible, but probably less efficient, to only validate the data within the managed computer.

Figure 11 schematically illustrates a general purpose computer 200 of the type that may be used to implement the above described techniques. The general purpose computer 200 includes a central processing unit 202, a random access memory 204, a read only memory 206, a network interface card 208, a hard disk drive 210, a display driver 212 and monitor 214 and a user input/output circuit 216 with a keyboard 218 and mouse 220 all connected via a common bus 222. In operation the central processing unit 202 will execute computer program instructions that may be stored in one or more of the random access memory 204, the read only memory 206 and the hard disk drive 210 or dynamically downloaded via the network interface card 208. The results of the processing performed may be displayed to a user via the display driver 212 and the monitor 214. User inputs for controlling the operation of the general purpose computer 200 may be received via the user input output circuit 216 from the keyboard 218 or the mouse 220. It will be appreciated that the computer program could be written in a variety of different computer languages. The computer program may be stored and distributed on a recording medium or dynamically downloaded to the general purpose computer 200. When operating under control of an appropriate computer program, the general purpose computer 200 can perform the above described techniques and can be considered to form an apparatus for performing the above described technique. The architecture of the general purpose computer 200 could vary considerably and Figure 11 is only one example.

Although illustrative embodiments of the invention have been described in detail herein with reference to the accompanying drawings, it is to be understood that the invention is not limited to those precise embodiments, and that various changes and modifications can be effected therein by one skilled in the art without departing from the scope and spirit of the invention as defined by the appended claims.

I CLAIM:

1. A computer program product for controlling a computer to validate program configuration data, said computer program product comprising:

comparing code operable to compare an XML data representation of said program configuration data with data defining valid program configuration data;

wherein, if said XML data representation does match said data defining valid program configuration data, then triggering code is operable to trigger a valid program configuration response.

2. A computer program product as claimed in claim 1, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program;

program initialisation data specifying configuration parameters of an application program; and

XML data directly specifying configuration parameters of an application program.

3. A computer program product as claimed in claim 1, comprising mapping code operable to map between said program configuration data and an XML data representation of said program configuration data.

4. A computer program product as claimed in claim 1, wherein said data defining valid program configuration data is one of:

XSD data defining a valid XML data representation; and

DTD data defining a valid XML data representation.

5. A computer program product as claimed in claim 1, wherein said comparing code is part of an XML parser.

6. A computer program product as claimed in claim 5, wherein said XML parser also provides validation of XML data other than said XML data representation of said program configuration data.

7. A computer program product as claimed in claim 3, wherein said mapping code is operable to map between said program configuration data and said XML data representation via a DOM data representation of said program configuration data.

8. A computer program product as claimed in claim 7, comprising editing code operable to edit said DOM data representation of said program configuration to provide modified program configuration to be validated.

9. A computer program product as claimed in claim 1, wherein said comparing code is executable by a program configuration managing computer and said valid program configuration response comprises sending validated program configuration data to a managed computer for use by said managed computer.

10. A computer program product as claimed in claim 9, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

11. A computer program product as claimed in claim 1, wherein said comparing code is executable by a managed computer which receives program configuration data from a program configuration managing computer and said valid program configuration response comprises configuring a program on said managed computer using said validated program configuration data.

12. A computer program product as claimed in claim 11, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

13. A computer program product for providing program configuration data for a computer, said computer program product comprising:

receiving code operable to receive an XML data representation of said program configuration data at said computer;

mapping code operable to map between said XML data representation and said program configuration data; and

configuration code operable to at least one of apply said program configuration data to said computer and retrieve said program configuration data from said computer.

14. A computer program product as claimed in claim 13, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program; and

program initialisation data specifying configuration parameters of an application program.

15. A computer program product as claimed in claim 13, wherein mapping between said XML data representation and said program configuration data is via a DOM data representation of said program configuration data.

16. A method of validating program configuration data, said method comprising the step of:

comparing an XML data representation of said program configuration data with data defining valid program configuration data;

wherein, if said XML data representation does match said data defining valid program configuration data, then triggering a valid program configuration response.

17. A method as claimed in claim 16, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program;

program initialisation data specifying configuration parameters of an application program; and

XML data directly specifying configuration parameters of an application program.

18. A method as claimed in claim 16, comprising the step of mapping between said program configuration data and an XML data representation of said program configuration data.

19. A method as claimed in claim 16, wherein said data defining valid program configuration data is one of:

XSD data defining a valid XML data representation; and

DTD data defining a valid XML data representation.

20. A method as claimed in claim 16, wherein said step of comparing is performed by an XML parser.

21. A method as claimed in claim 20, wherein said XML parser also provides validation of XML data other than said XML data representation of said program configuration data.

22. A method as claimed in claim 18, wherein mapping between said program configuration data and said XML data representation is via a DOM data representation of said program configuration data.

23. A method as claimed in claim 22, comprising the step of editing said DOM data representation of said program configuration to provide modified program configuration to be validated.

24. A method as claimed in claim 16, wherein said step of comparing is performed by a program configuration managing computer and said valid program configuration response comprises sending validated program configuration data to a managed computer for use by said managed computer.

25. A method as claimed in claim 24, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

26. A method as claimed in claim 16, wherein said step of comparing is performed by a managed computer which receives program configuration data from a program configuration managing computer and said valid program configuration response

comprises configuring a program on said managed computer using said validated program configuration data.

27. A method as claimed in claim 26, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

28. A method of providing program configuration data for a computer, said method comprising the steps of:

receiving an XML data representation of said program configuration data at said computer;

mapping between said XML data representation and said program configuration data; and

at least one of applying said program configuration data to said computer and retrieving said program configuration data from said computer.

29. A method as claimed in claim 28, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program; and

program initialisation data specifying configuration parameters of an application program.

30. A method as claimed in claim 28, wherein mapping between said XML data representation and said program configuration data is via a DOM data representation of said program configuration data.

31. Apparatus for validating program configuration data, said apparatus comprising:

comparing logic operable to compare an XML data representation of said program configuration data with data defining valid program configuration data;

wherein, if said XML data representation does match said data defining valid program configuration data, then triggering logic is operable to trigger a valid program configuration response.

32. Apparatus as claimed in claim 31, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program;

program initialisation data specifying configuration parameters of an application program; and

XML data directly specifying configuration parameters of an application program.

33. Apparatus as claimed in claim 31, comprising mapping logic operable to map between said program configuration data and an XML data representation of said program configuration data.

34. Apparatus as claimed in claim 31, wherein said data defining valid program configuration data is one of:

XSD data defining a valid XML data representation; and

DTD data defining a valid XML data representation.

35. Apparatus as claimed in claim 31, wherein said comparing logic is part of XML parsing logic.

36. Apparatus as claimed in claim 35, wherein said XML parsing logic also provides validation of XML data other than said XML data representation of said program configuration data.

37. Apparatus as claimed in claim 33, wherein said mapping logic is operable to map between said program configuration data and said XML data representation via a DOM data representation of said program configuration data.

38. Apparatus as claimed in claim 37, comprising editing logic operable to edit said DOM data representation of said program configuration to provide modified program configuration to be validated.

39. Apparatus as claimed in claim 31, wherein said comparing logic is part of a program configuration managing computer and said valid program configuration response comprises sending validated program configuration data to a managed computer for use by said managed computer.

40. Apparatus as claimed in claim 39, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

41. Apparatus as claimed in claim 31, wherein said comparing logic is part of a managed computer which receives program configuration data from a program configuration managing computer and said valid program configuration response comprises configuring a program on said managed computer using said validated program configuration data.

42. Apparatus as claimed in claim 41, wherein said validated program configuration data is sent from said program configuration managing computer to said managed computer as said XML data representation.

43. Apparatus for providing program configuration data for a computer, said apparatus comprising:

receiving logic operable to receive an XML data representation of said program configuration data at said computer;

mapping logic operable to map between said XML data representation and said program configuration data; and

configuration logic operable to at least one of apply said program configuration data to said computer and retrieve said program configuration data from said computer.

44. Apparatus as claimed in claim 43, wherein said program configuration data is one of:

operating system registry data specifying configuration parameters of an application program; and

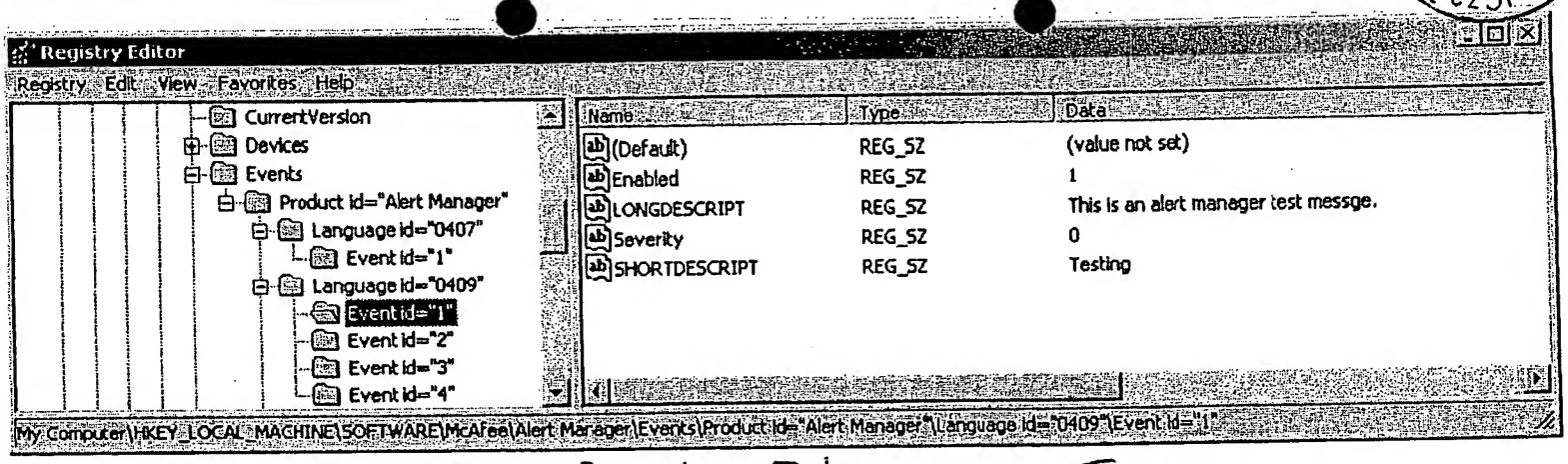
program initialisation data specifying configuration parameters of an application program.

45. Apparatus as claimed in claim 43, wherein mapping between said XML data representation and said program configuration data is via a DOM data representation of said program configuration data.

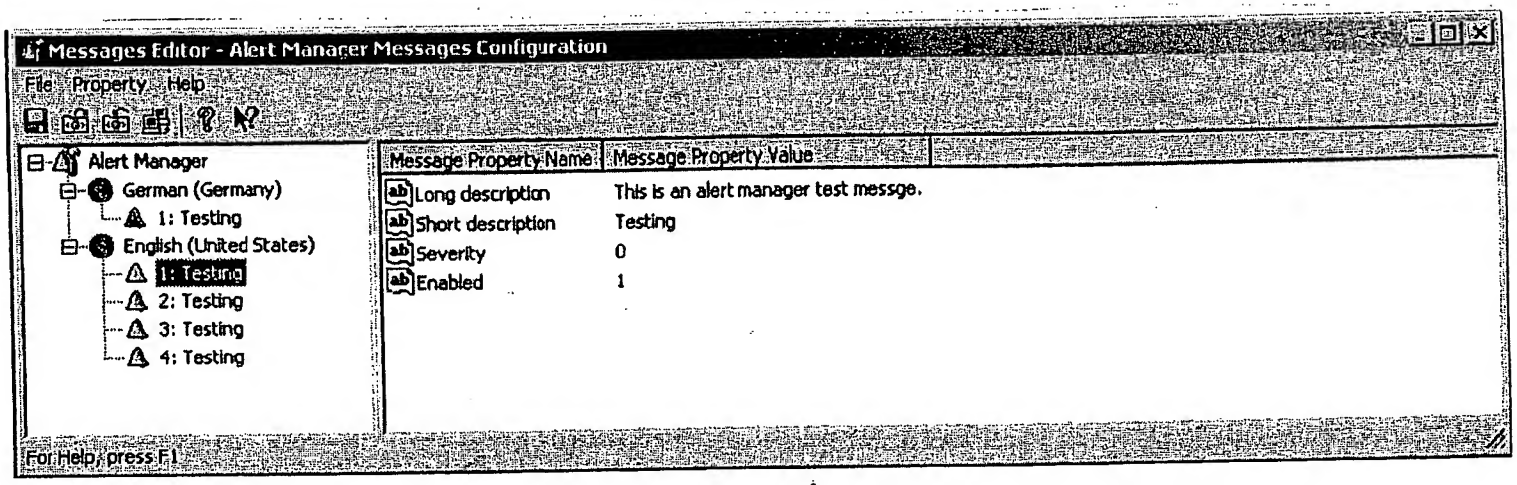
ABSTRACT OF THE DISCLOSURE

Computer program configuration data, which may be in the form of Windows Registry data, is mapped into an XML data representation of that configuration data. This XML data representation is then compared with an XSD data template defining valid configuration parameters. The comparison may be performed by an XML parser. The technique may be used to validate configuration changes during editing, confirm that an application has a valid configuration and/or and to distribute initial configurations or modified configurations with a validation step provided to increase reliability.

[Figure 6]



Registry Data Fig. 1

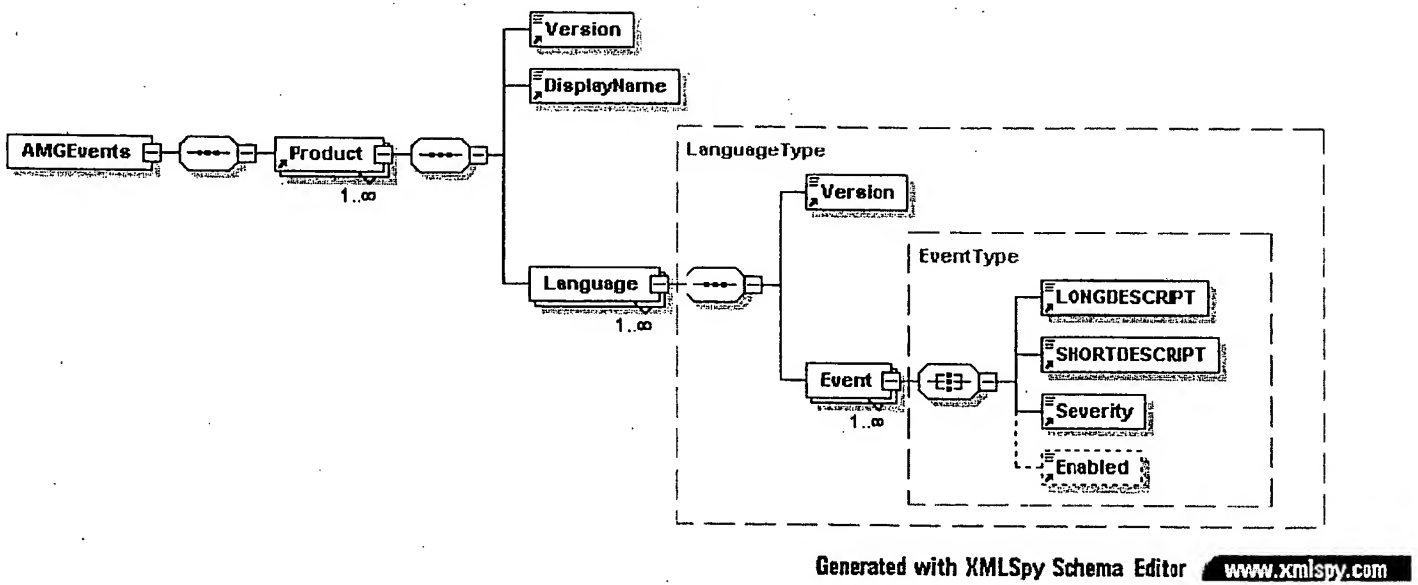


DOM Data View Fig. 2

```
<?xml version="1.0" ?>
- <AMGEvents xmlns="http://www.nai.com"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.nai.com AMGEvents.xsd">
  - <Product id="Alert Manager">
      <Version>0x04070000</Version>
      <DisplayName>Alert Manager 4.7</DisplayName>
    - <Language id="0407">
        <Version>0x01000002</Version>
      - <Event id="1">
          <LONGDESCRIPT>Das ist eine Test-Nachricht von Alert
            Manager.</LONGDESCRIPT>
          <SHORTDESCRIPT>Testing</SHORTDESCRIPT>
          <Severity>5</Severity>
          <Enabled>1</Enabled>
        </Event>
      </Language>
    - <Language id="0409">
        <Version>0x01000002</Version>
      - <Event id="1">
          <LONGDESCRIPT>This is an alert manager test
            messge.</LONGDESCRIPT>
          <SHORTDESCRIPT>Testing</SHORTDESCRIPT>
          <Severity>0</Severity>
          <Enabled>1</Enabled>
        </Event>
      - <Event id="2">
          <LONGDESCRIPT>Text of event 2.</LONGDESCRIPT>
          <SHORTDESCRIPT>Testing</SHORTDESCRIPT>
          <Severity>1</Severity>
        </Event>
      - <Event id="3">
          <LONGDESCRIPT>Text of event 3.</LONGDESCRIPT>
          <SHORTDESCRIPT>Testing</SHORTDESCRIPT>
          <Severity>1</Severity>
        </Event>
      - <Event id="4">
          <LONGDESCRIPT>Text of event 4.</LONGDESCRIPT>
          <SHORTDESCRIPT>Testing</SHORTDESCRIPT>
          <Severity>1</Severity>
        </Event>
      </Language>
  </Product>
</AMGEvents>
```

XML Data

Fig. 3



XSD Data

Fig. 4

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- edited with XML Spy v4.0.1 U (http://www.xmlspy.com) by Napalm
(Napalm)  -->
- <xs:schema targetNamespace="http://www.nai.com"
    xmlns="http://www.nai.com"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <xs:element name="DisplayName" type="xs:string" />
    <xs:element name="Enabled" type="xs:boolean" />
  - <xs:complexType name="EventType">
    - <xs:all>
        <xs:element ref="LONGDESCRIPT" />
        <xs:element ref="SHORTDESCRIPT" />
        <xs:element ref="Severity" />
        <xs:element ref="Enabled" minOccurs="0" />
      </xs:all>
      <xs:attribute name="id" type="xs:string" use="required" />
    </xs:complexType>
  - <xs:complexType name="LanguageType">
    - <xs:sequence>
        <xs:element ref="Version" />
        <xs:element name="Event" type="EventType"
          maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="id" type="xs:string" use="required" />
    </xs:complexType>
  - <xs:element name="Product">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="Version" />
          <xs:element ref="DisplayName" />
          <xs:element name="Language" type="LanguageType"
            maxOccurs="unbounded" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required" />
      </xs:complexType>
    </xs:element>
  - <xs:element name="AMGEvents">
    - <xs:complexType>
      - <xs:sequence>
          <xs:element ref="Product" maxOccurs="unbounded" />
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="LONGDESCRIPT" type="xs:string" />
    <xs:element name="SHORTDESCRIPT" type="xs:string" />
    <xs:element name="Severity" type="xs:string" />
    <xs:element name="Version" type="xs:string" />
</xs:schema>
```

XSD Data

Fig. 5

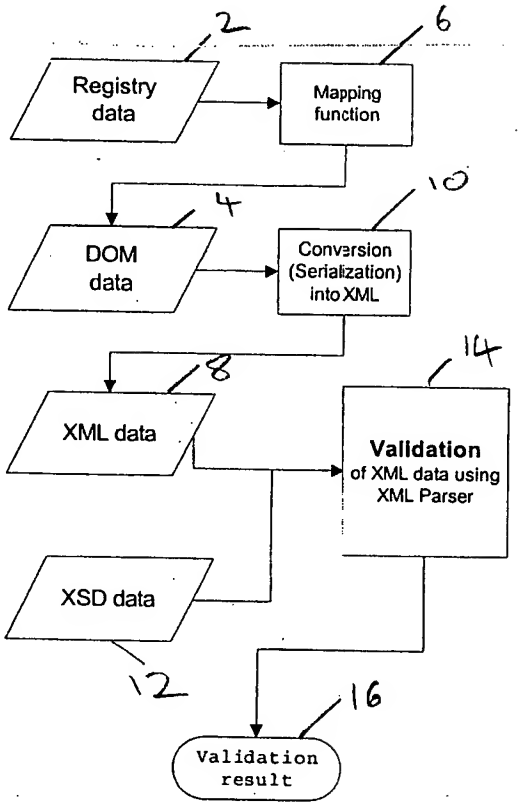


Fig. 6

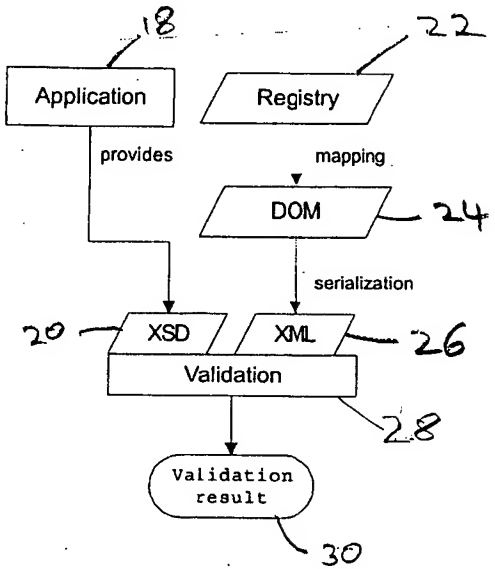


Fig. 7

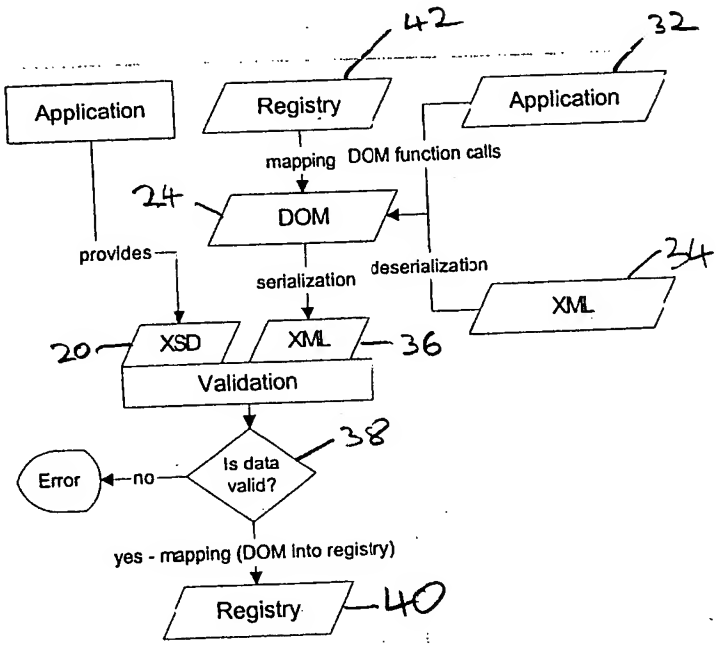


Fig. 8

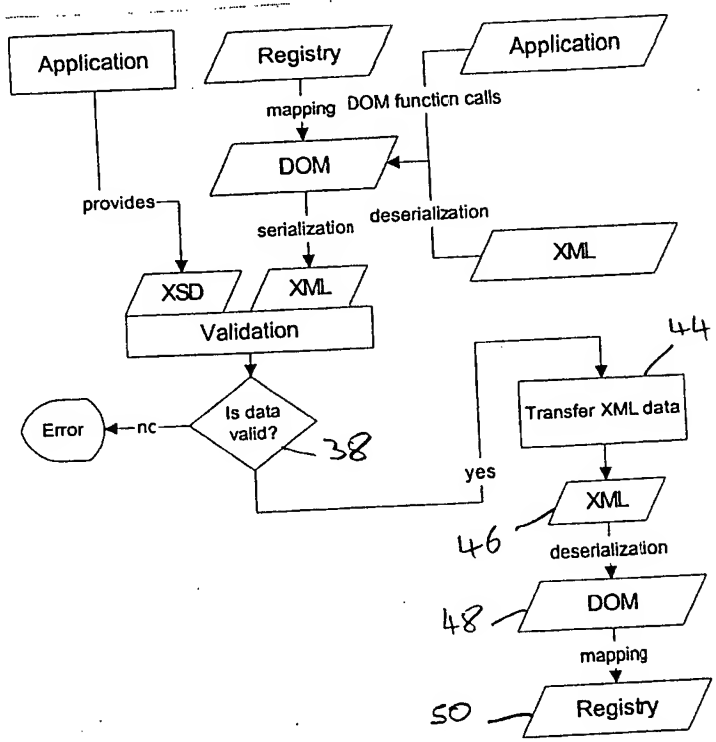


Fig. 9

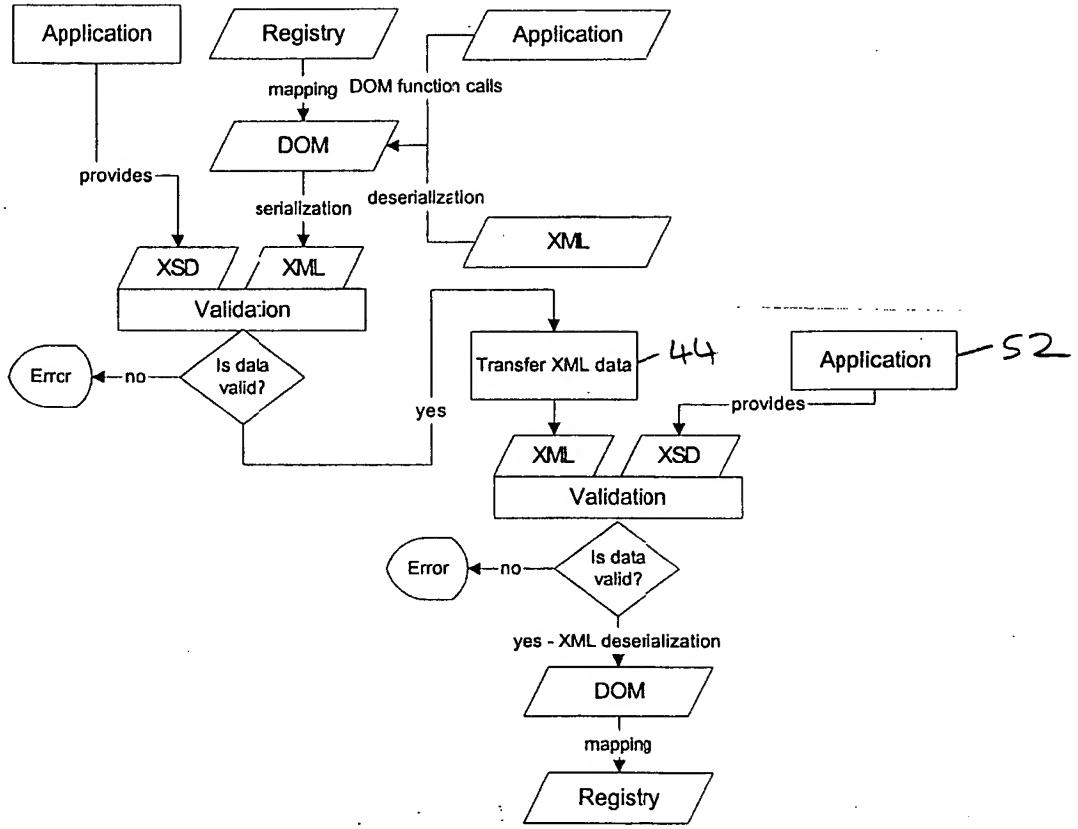

Application
Registry
Application
mapping
DOM function calls
DOM
provides
serialization
deserialization
XML
XSD
XML
Validation
Error
no
Is data valid?
yes
Transfer XML data
44
Application
52
provides
XML
XSD
Validation
Error
no
Is data valid?
yes - XML deserialization
DOM
mapping
Registry

Fig. 10

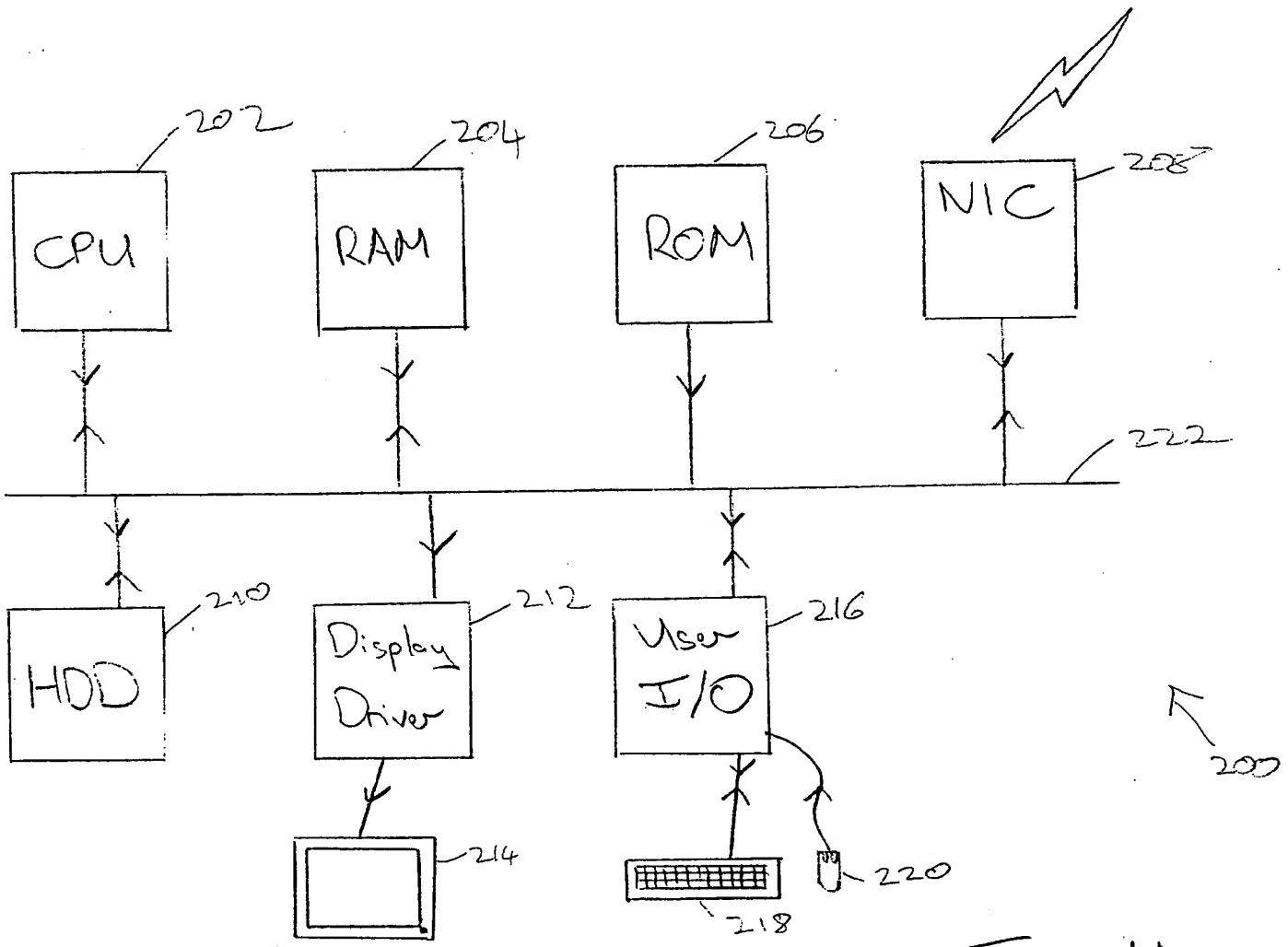

202
CPU
204
RAM
206
ROM
208
NIC
222
210
HDD
212
Display Driver
216
User I/O
214
218
220
200

Fig. 11